PARALLEL PERFORMANCE INVESTIGATIONS OF AN UNSTRUCTURED MULTIGRID SOLVER

D. J. Mavriplis

Institute for Computer Applications in Science and Engineering (ICASE)
NASA Langley Research Center
Hampton, VA

Co-PI, ASCI Level 2 Site
Old Dominion University
Norfolk, VA

ASCI Solvers Workshop

December 1-2, 1999, Los Alamos, NM

MOTIVATION

- Develop Large-Scale Simulation Capability using Unstructured Multigrid Solver
  - Large-Eddy Simulation (up to 10^9 Grid Points)
  - Radiation Transport Solver (Diffusion Approximation)
- Implement Combined MPI-OMP Domain Based Parallelization Strategy
  - Suitable for Hybrid Shared-Distributed Memory Systems
- Benchmark on Currently Available Architectures
- Evaluate New Architectures as they become Available

OVERVIEW

- Governing Equations, Discretization
- Multigrid Solution Algorithm
  - Agglomeration
  - Anisotropic
- Combined MPI/OMP Parallelization
- Benchmark Results
  - Up to 2048 Processors
  - Up to 25 million points, 125 million unknowns

BASE SOLVER

- Governing Equations : Reynolds-Averaged Navier-Stokes
  - Conservation of Mass, Momentum, Energy
  - Single Equation Turbulence Model (Spalart-Allmaras)
    - Convection-Diffusion-Production
- Vertex-Based Discretization
  - 2nd order upwind finite-volume scheme
  - 6 variables per grid point
  - Flow equation fully coupled (5×5)
  - Turbulence equation uncoupled

BASE SOLVER

- Mixed Element Grids
  - Tetrahedra, Prisms
  - Pyramids, Hexahedra
- Edge Data-Structure
  - Building Block for All Element Types
  - Lower Memory Overheads
  - Higher Computational Rates
- Explicit Multi-Stage Time-Stepping (Preconditioned)

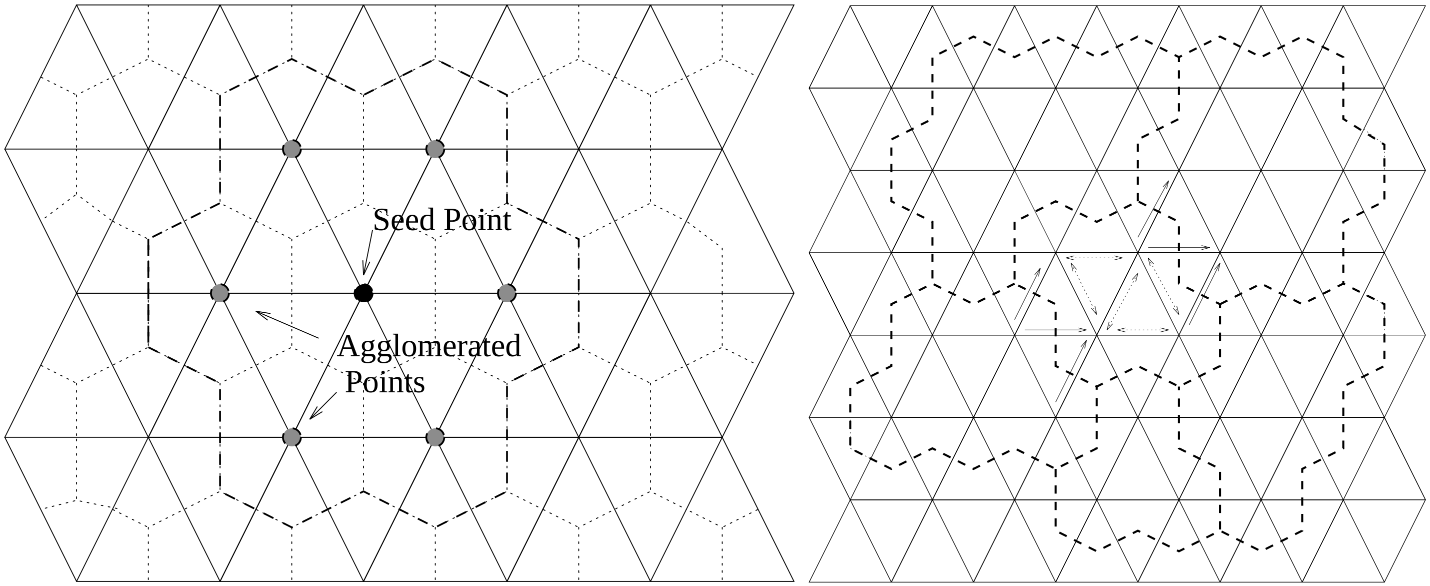
CONVERGENCE ACCELERATION

- Agglomeration Multigrid
  - Automatic Coarse Level Construction
- Line-Implicit Solver / Jacobi Preconditioning
- Low Mach Number Preconditioning
- Non Linear GMRES (using above solver as a preconditioner)

AGGLOMERATION MULTIGRID

- Principal Convergence Acceleration Ingredient
- Grid Independent Convergence Rates
- Low Memory Overheads
  - No Explicit Linearization (FAS)
- Latency Tolerant
  - Based on Sequence of Coarse-Fine Grids
  - Explicit (or Locally Implicit) Solver on Each Grid Level

AGGLOMERATION MULTIGRID (Non-Linear Problems)



- Merge Control Volumes to Form Coarse Levels
  - Graph-Based AMG Coarsening
- Transfer between Grid Levels via Piecewise Constants
- Coarse Level Eqns obtained by Summation of Fine Level Eqns
  - Algebraic summation of solution independent terms
  - Restriction of Solution Dependent Terms (FAS)

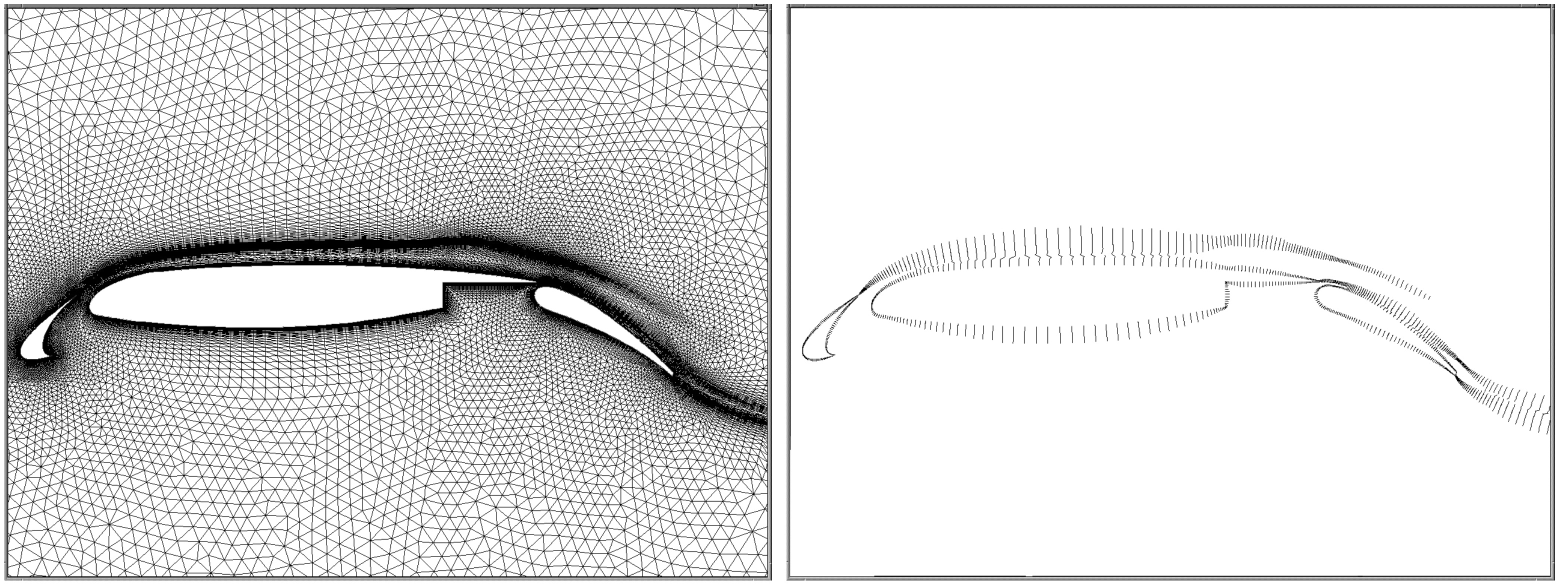
ANISOTROPY-INDUCED STIFFNESS

- Convergence Rates for High-Reynolds Number Flows Much Slower
  - Mainly Due to Grid Stretching $\approx O(10^4)$

- Standard Techniques for Anisotropic Problems
  - Directional (Semi) Coarsening Multigrid
  - Directional (Line) Solvers

IMPLICIT LINE PRECONDITIONING

- Graph Algorithm Used to Construct Lines in Regions of High Grid Stretching
- Implicit System Solved Along Lines
- Reduces to Jacobi Preconditioning in Isotropic Regions

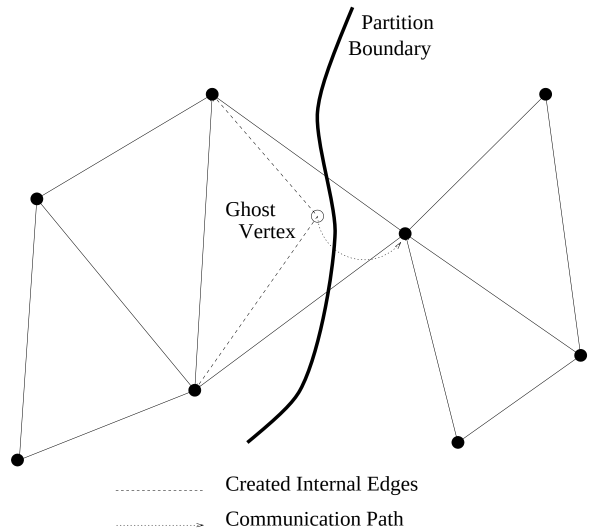


NON-LINEAR GMRES ALGORITHM

- Preconditioned Multigrid Algorithm May be Used as Preconditioner to Non-Linear GMRES
- Potential Speedup in Convergence
- Incurs Additional Memory Overheads (Storage of Search Directions)
  - GMRES (20) requires $\approx 50\%$ increase in Memory
- Simple Parallelization Strategy
  - Non-Linear Function Evaluations Already Parallelized
  - Low Order (20) Least Squares Problem Performed Redundantly on Each Processor

PARALLEL IMPLEMENTATION

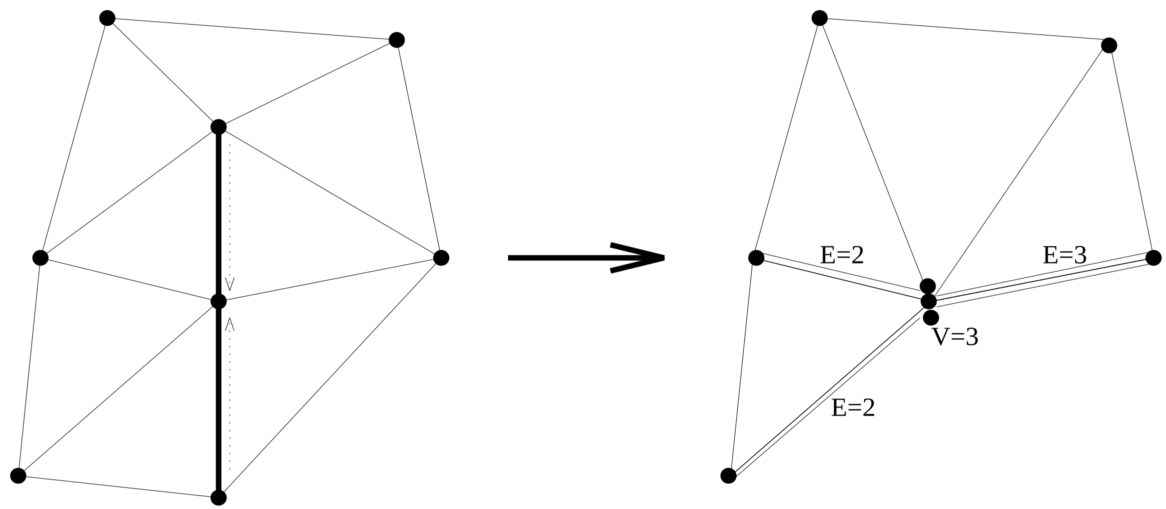
- Domain Decomposition using MPI and/or OpenMP
  - Portable, Distributed and Shared Memory Architectures



- Weighted Partitioning to Avoid Intersected Line Edges
  - CHACO, MeTiS
- Coarse and Fine Multigrid Levels Partitioned Independently

PARTITIONING

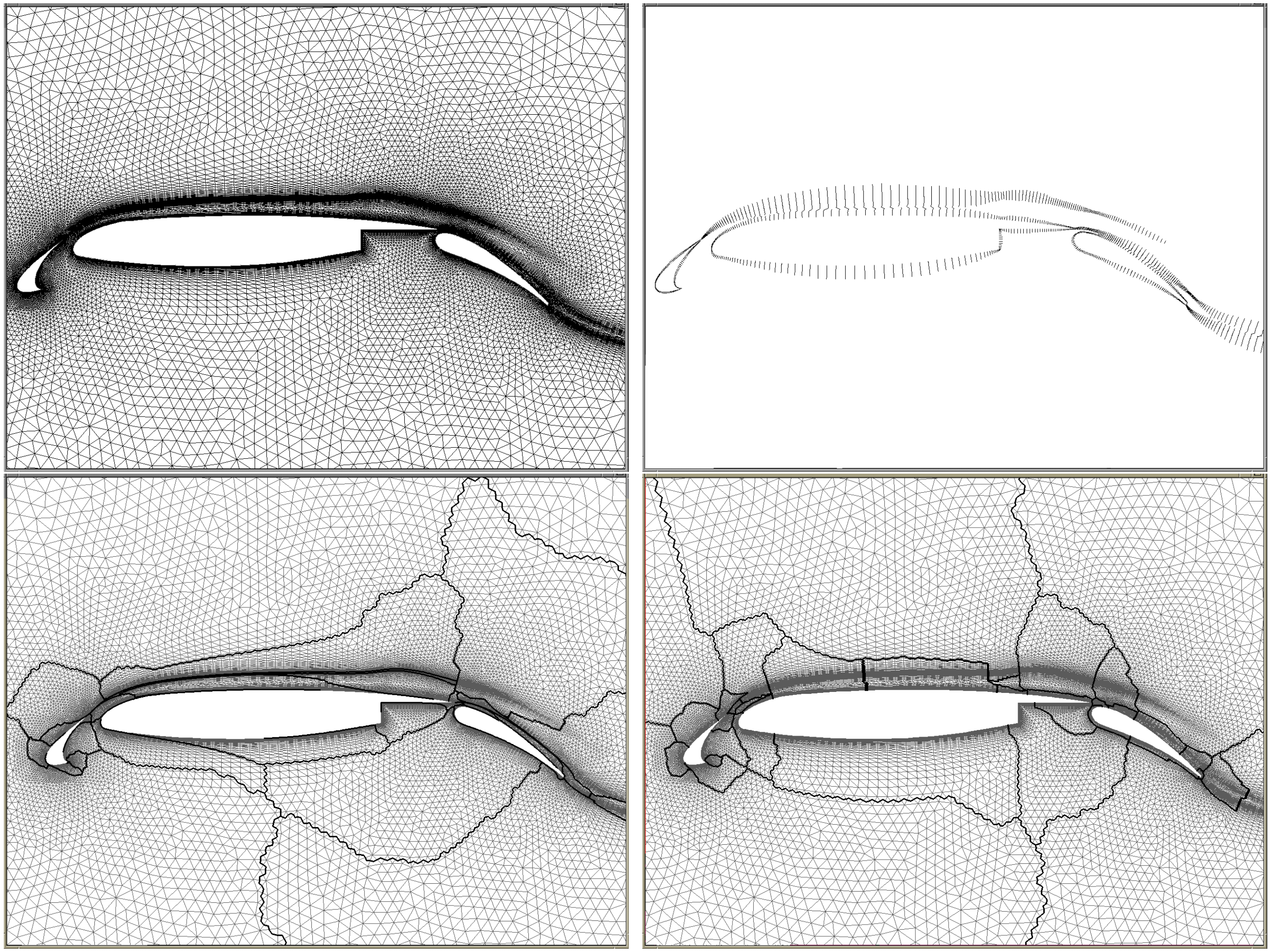
- Contract Graph Along Implicit Lines
- Weight Edges and Vertices



- Partition Contracted Graph
- Decontract Graph
  - Guarantees Lines Never Broken
  - Possible Small Increase in Imbalance/Cut Edges

PARTITIONING EXAMPLE

- 32-Way Partition of 30,562 Point 2D Grid



- Unweighted Partition: 2.6 % Edges Cut, 2.6 % Lines Cut
- Weighted Partition: 3.2 % Edges Cut, 0 % Lines Cut

PARTITIONING FOR MULTIGRID

- Partition Fine Grid Level
- Partition Coarse AMG Level Graphs
- Nested Levels
  - Fine Level Partition Could Be Used to Infer Coarse Level Partitions
  - Optimizes Inter-Level Communication
- Partition Levels Independently
  - Optimize Intra-Level Communication
  - Heuristic Procedure to Match Coarse/Fine Level Partitions

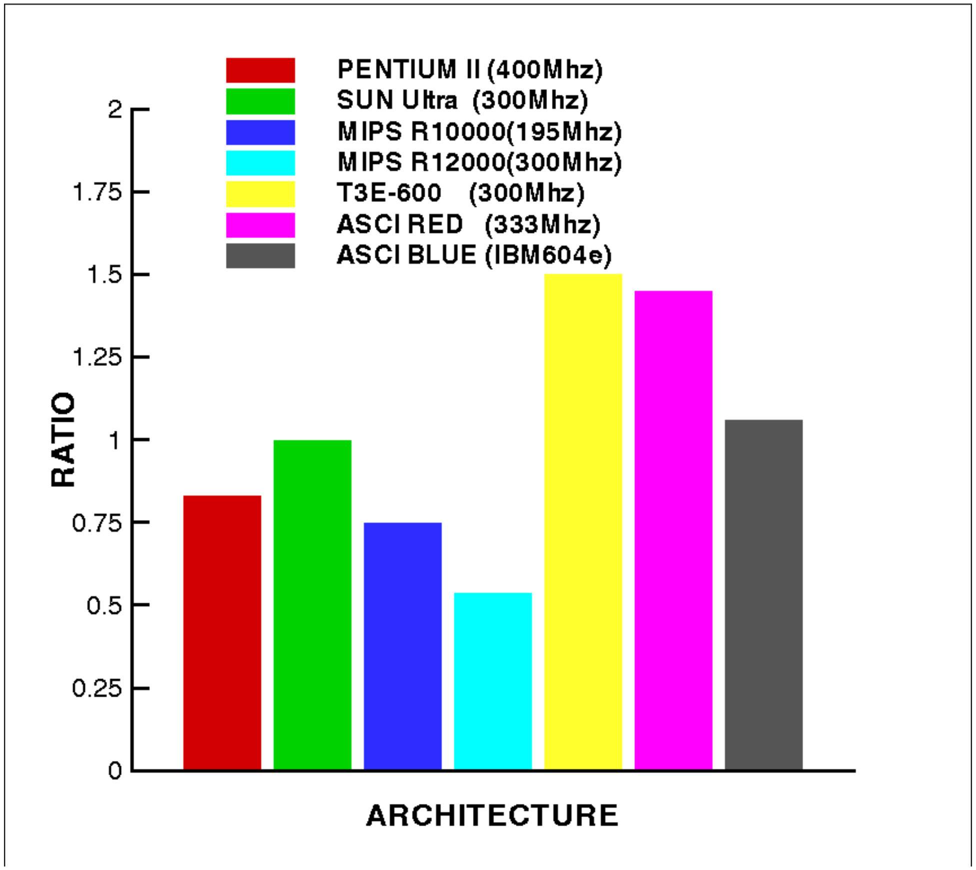
PRE-PROCESSING REQUIREMENTS

- Pre-processing Operations
  - Construction of Coarse AMG Levels
  - Construction of Implicit Lines
  - Partitioning of Mesh Levels
- Minimal CPU-Time Requirements
- Memory Requirements Comparable to Flow solver
- Considerable Logic for Parallelization
- Benefits of Shared-Memory Paradigm
  - Run Sequentially
  - Access Large Amounts of Off-processor memory
- Will Eventually Require Parallelization

SINGLE PROCESSOR OPTIMIZATIONS

- Scalar Microprocessors
  - Vertices and Edges Reordered for Locality
  - RCM-type algorithm : Factor 2 speedup on Sun Workstation
- Vector Processors
  - Vertices Reordered for Locality
  - Edges Sorted into Non-Recurrent (vectorizable) Groups
  - Line Solves performed in (vector) Groups of 64
- Sample Performance (1 grid level)
  - MIPS R10000 (250Mhz): 75 Mflops
  - Pentium II (400Mhz): 50 Mflops
  - Cray SV1 Vector Processor: 225 Mflops

RELATIVE EXECUTION TIME OF VARIOUS MICRO-PROCESSORS



- 3D Unstructured Multigrid Algorithm on 177K Grid

PARALLEL PROGRAMMING MODELS

- MPI: Distributed Memory
- OpenMP: Shared Memory
- Mixed Model: Clusters of Shared-Memory Multiprocessors
  - Dual CPU Pentium Clusters
  - ASCI Machines
- cc-NUMA Architecture (SGI Origin)
  - Logically Shared
  - Physically Distributed

EXTENDING MPI CODE TO MIXED MPI-OpenMP MODEL

- MPI Process Rewritten to Handle Multiple Domains
  - Sequentially
  - In Parallel Using OpenMP
- Flexibility
  - Run MPI or OpenMP Exclusively
  - Run Two-Level MPI-OpenMP Model
  - Sequential Capability
    - Number of Domains can be Multiple of Number of Processors
- Entirely Domain-Based Parallelism

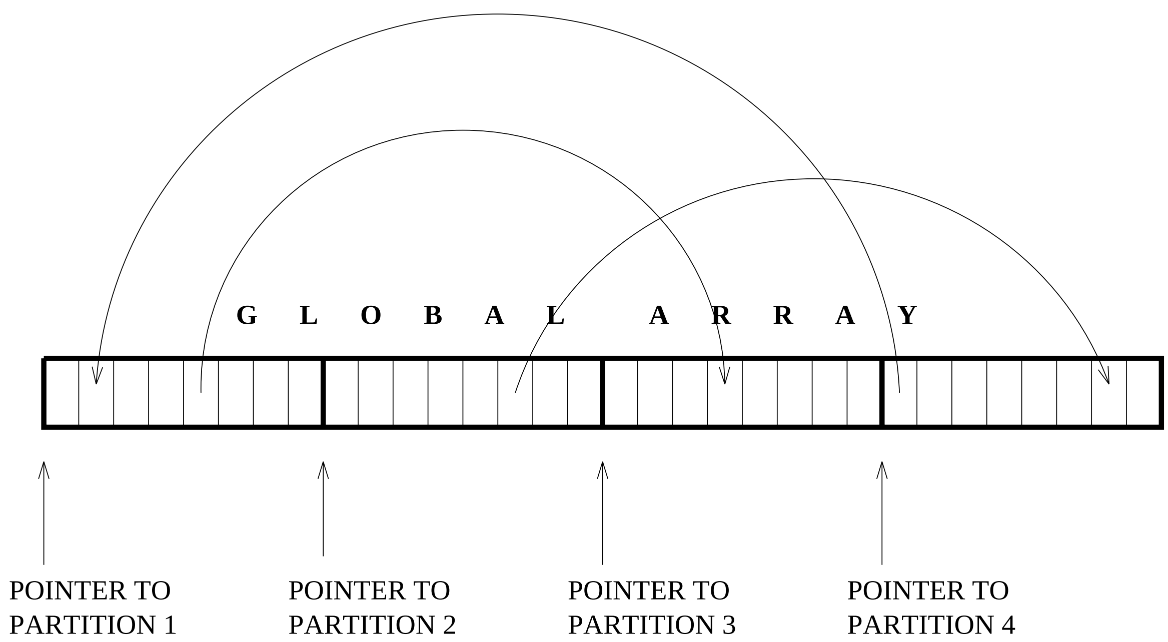
OVERALL CODE STRUCTURE

```
include OMP_DIRECTIVE
do : Loop over number of partitions
        do : Loop over number of vector groups
                do : Loop over edges in a vector group
                        n1 = edge_end(1,iedge)
                        n2 = edge_end(1,iedge)
                        flux = function of values at n1,n2
                        residual(n1) = residual(n1) + flux
                        residual(n2) = residual(n2) - flux
                enddo
        enddo
 enddo
c
include OMP_DIRECTIVE
do : Loop over number of partitions
        call OMP_communicate
enddo
c
include OMP_DIRECTIVE
do : Loop over number of partitions
        call MPI_communicate
enddo
```

- Entire Code OMP'ed with 2 or 3 Directives
- Distinct Partition Loops (instead of OMP BARRIER) enables Code to run Sequentially

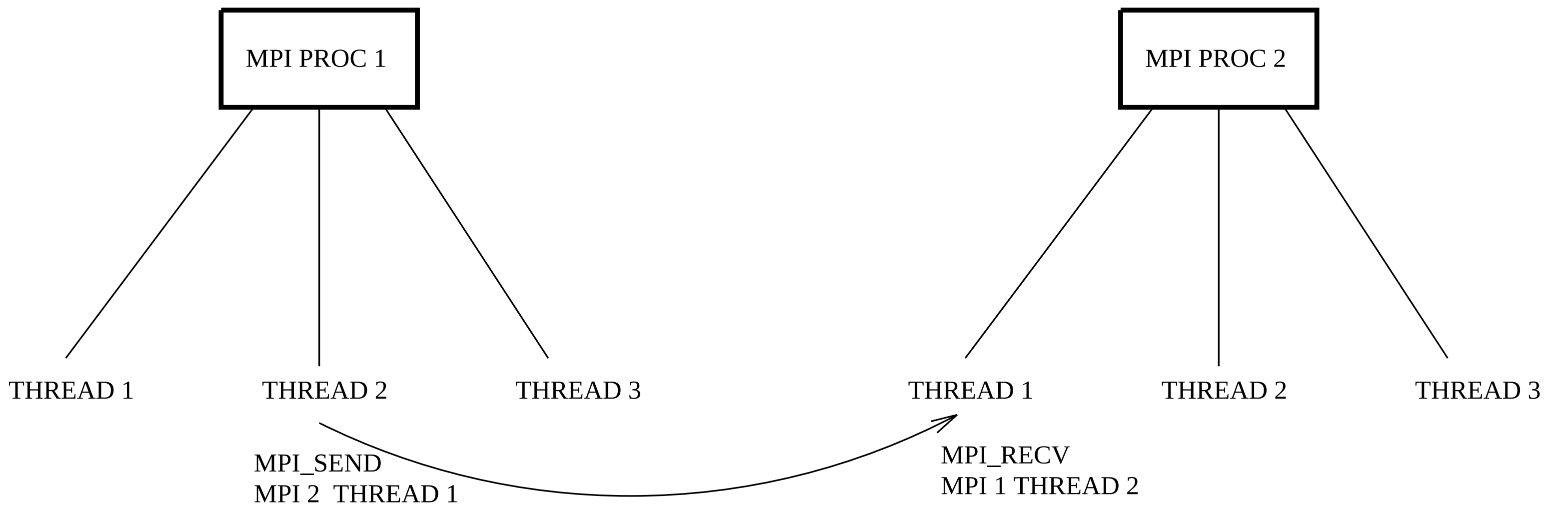
OPENMP COMMUNICATION (within an MPI Process)

- Arrays Span All Local Partitions/Threads
- Pointers used to Identify Extent of Each Partition/Thread
- Local Indices (relative to pointers) used in Computation Loop
- Global Indices Used for Communication
- Communicate by Copying Selected Values to Specific Locations in Global Array



COMMUNICATION BETWEEN MPI PROCESSES

- Thread to Thread MPI Messages
  - Each Thread Sends to/ Receives from:
    - An MPI Process
    - A Thread Id (implemented as message tag)
- Entirely Parallel Provided MPI Implementation is THREAD-SAFE



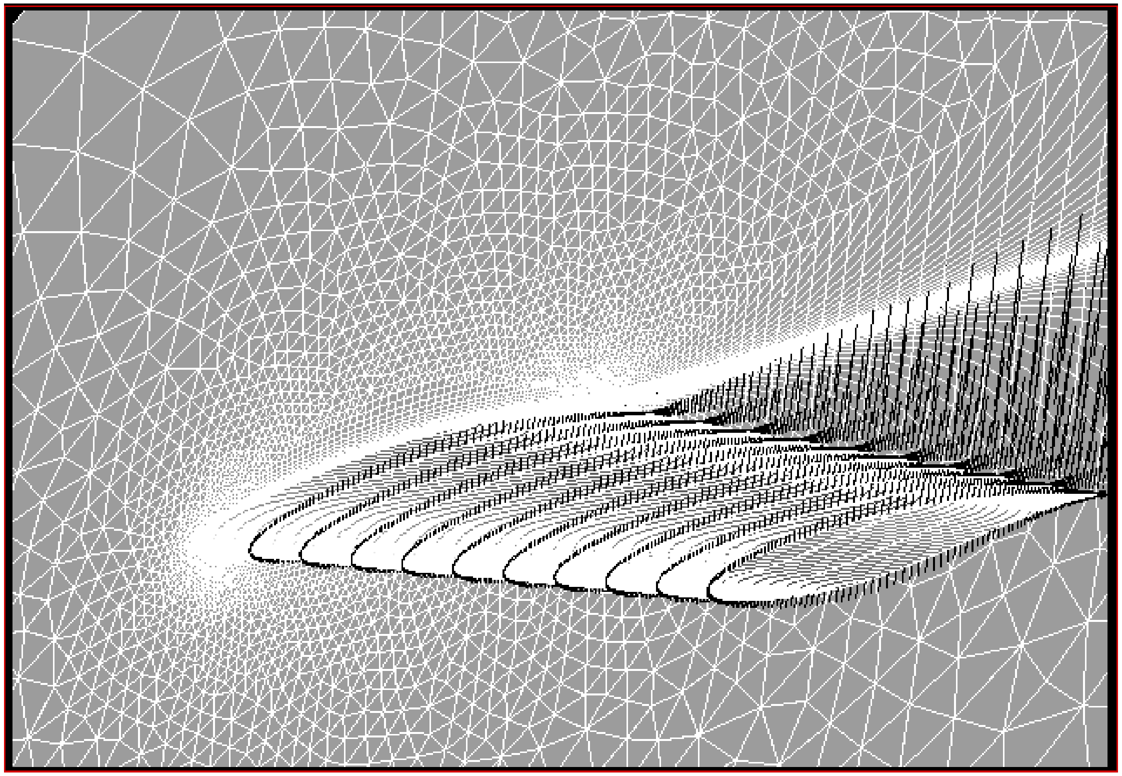
MIXED MODEL COMMUNICATION

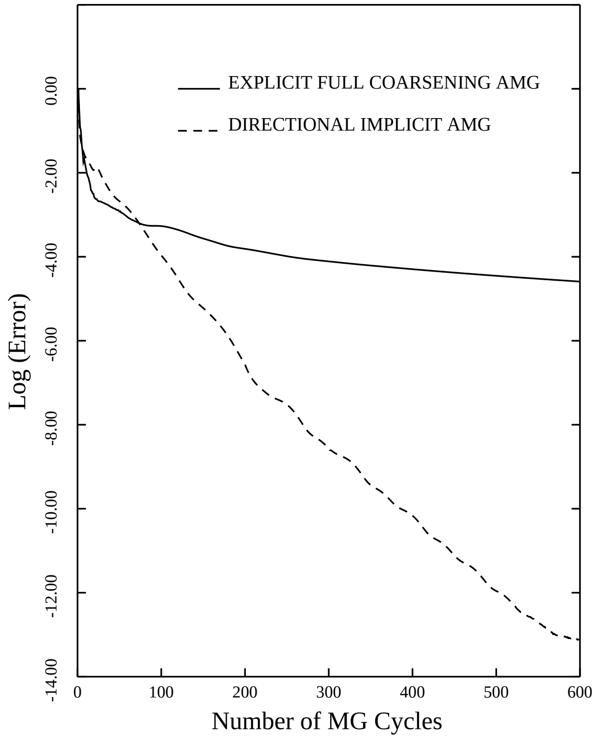
- MPI Communication Reduced by Intra-Process OMP Communication
- Partitions should be Mapped to:
  - Maximize Intra-Process OMP Communication
  - Minimize Inter-Process MPI Communication
- Output Weighted Communication Graph (between all Partitions)
  - Partition Communication Graph Using METIS/CHACO
  - Identifies Groupings of Partitions
- METIS Partitions Numbered Naturally for Locality
  - Simple Blocked Mapping of Metis Partition Numbering Produces Equivalent (or Better) Results to Explicit Partitioning of Communication Graph

PARALLEL SCALABILITY RESULTS

- MPI Alone on ASCI Machines, SGI O2K, T3E
- Comparison of MPI versus OpenMP Performance on Shared Memory Machines: SGI Origin, Cray SV1
- Mixed OpenMP/MPI on SGI Origin, Dual CPU Pentium Cluster
- Effect of Problem Size
  - 177,000 Point Problem
  - 3 million and $3M \times 8 = 24M$ Point Problems

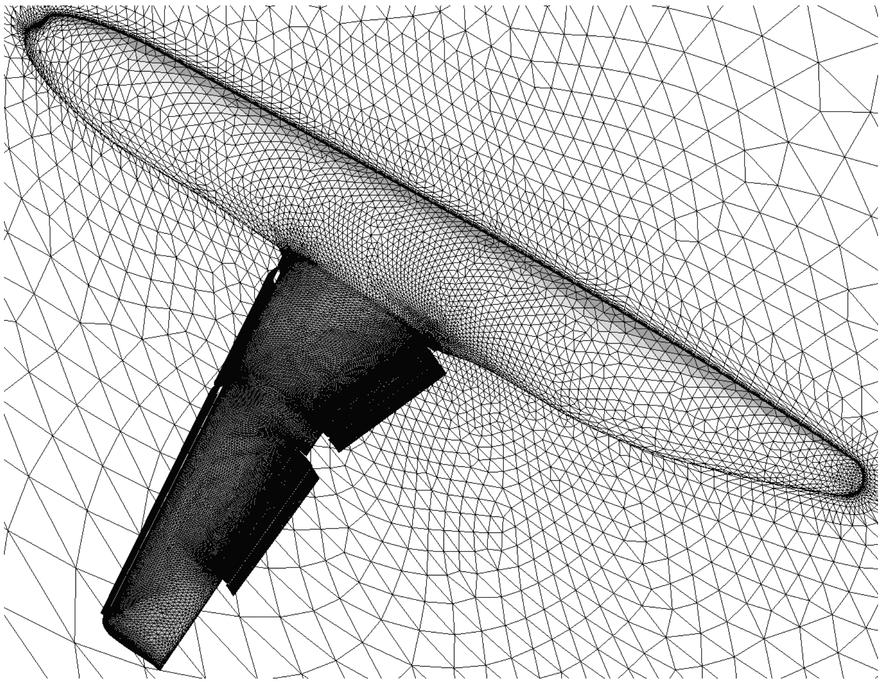
RAE-WING TEST-CASE

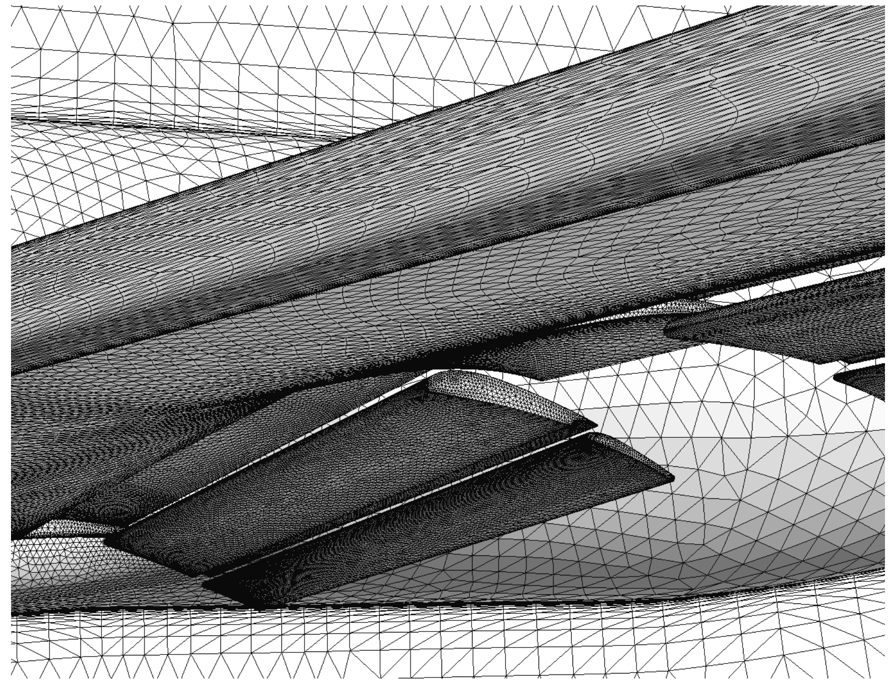




- 177,837 Vertices (Mixed Hexahedra and Prisms)
- 67 % Fine Grid Points Belong to Lines
- Order of Magnitude Faster than Isotropic Scheme
- Mach = 0.73, Incidence = 2.31 degrees, Reynolds = 6.5 million

FULL AIRCRAFT HIGH-LIFT CONFIGURATION

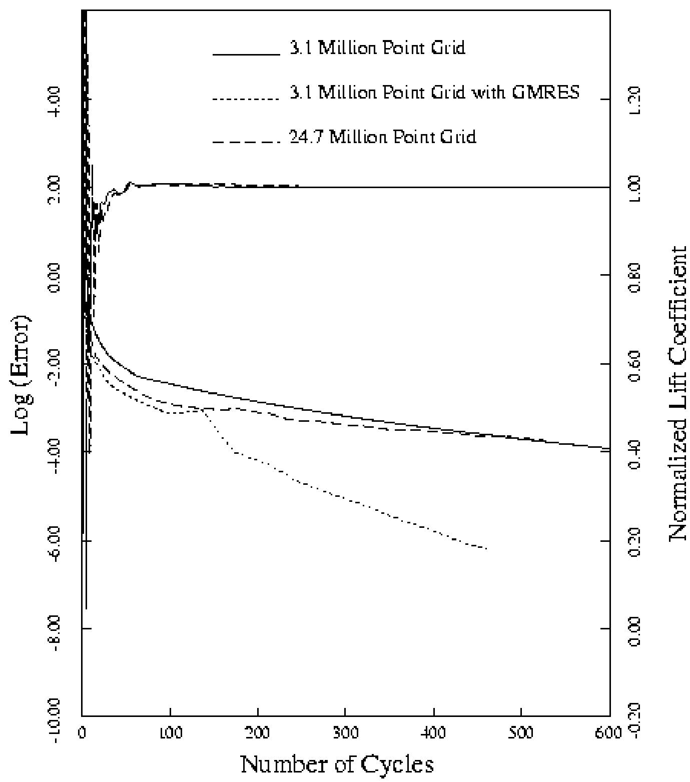




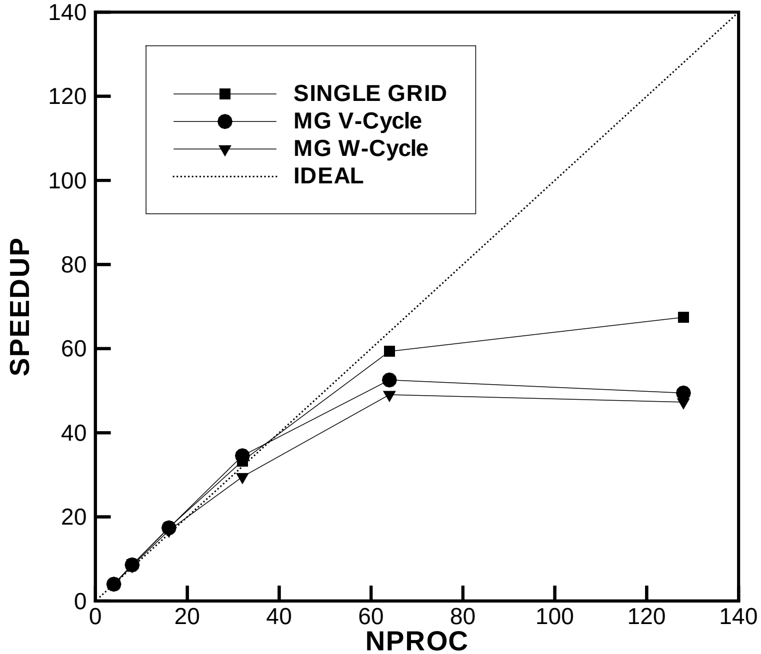
- Mixed Prismatic-Tetrahedral Mesh
- Fine Mesh: 3.1 million points, 18 million tetrahedra
- Coarse Mesh: 24.7 million points, 145 million tetrahedra

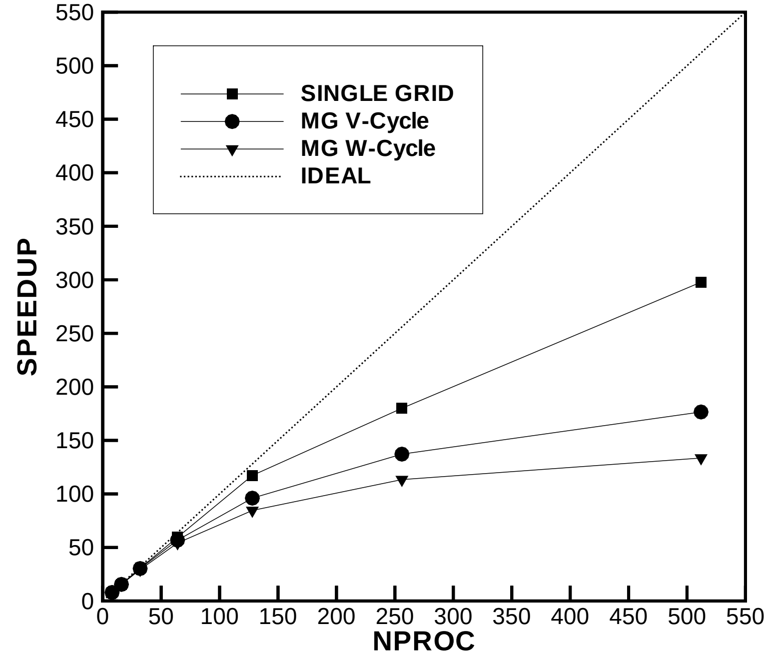
CONVERGENCE HISTORIES

- Coarse Mesh: 4 orders on 600 Multigrid Cycles
- Fine Mesh: Similar to Coarse Mesh
  - Grid Independence Property of Multigrid
- Beneficial Effects of GMRES for Coarse Grid
  - Insufficient Memory for GMRES on Fine Mesh



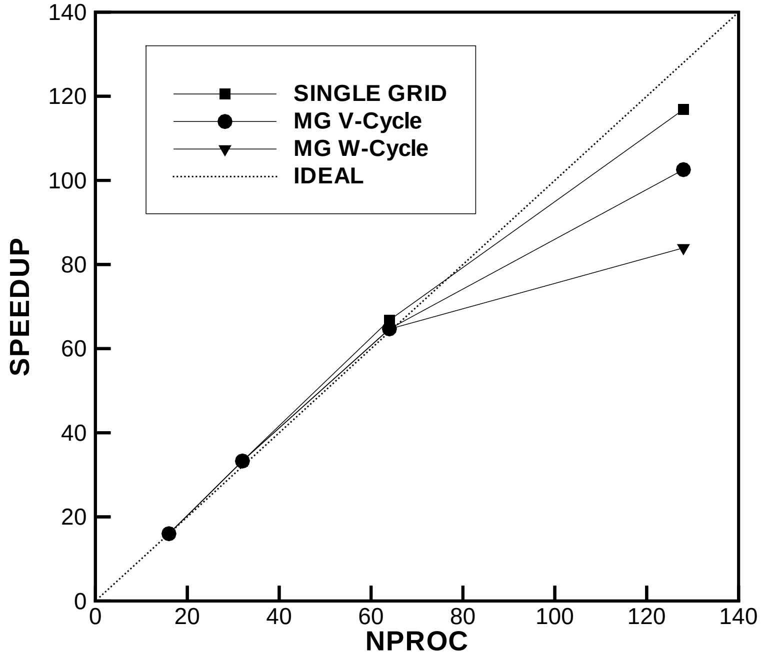
RAE WING SCALABILITY RESULTS

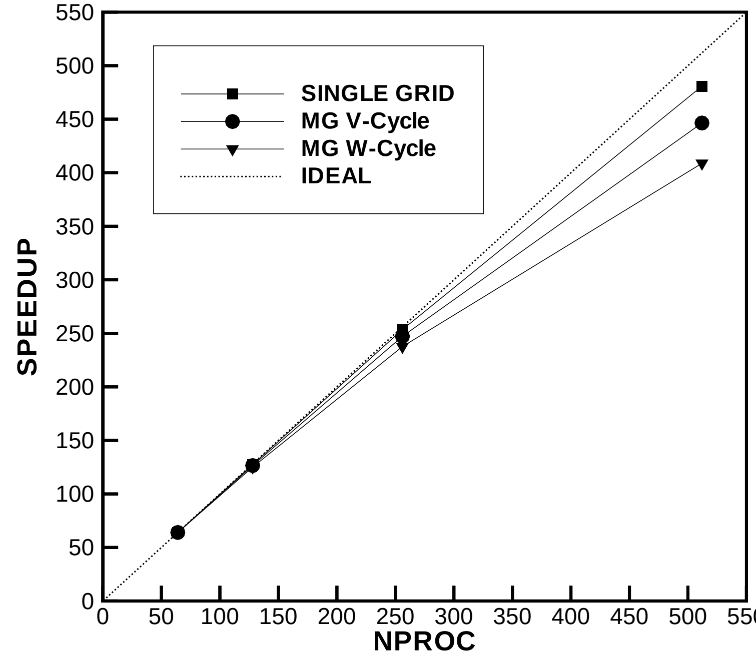




- Good Scalability up Moderate Number of Processors
- Increased Communication for MG Coarse Levels
  – Small Problem Size; On 512 Processors:
  – Fine Level: 348 points per processor
  – Coarse Level: 13 points per processor

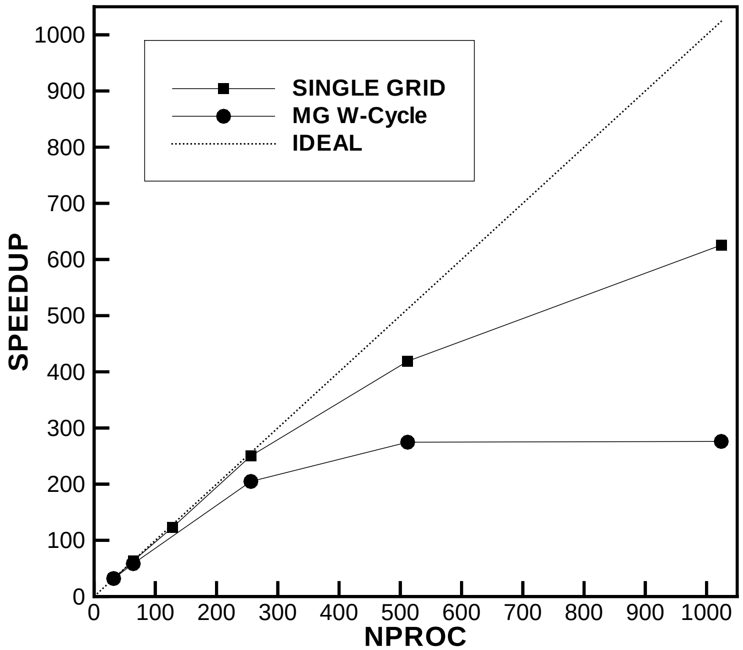
3 M POINT SCALABILITY RESULTS



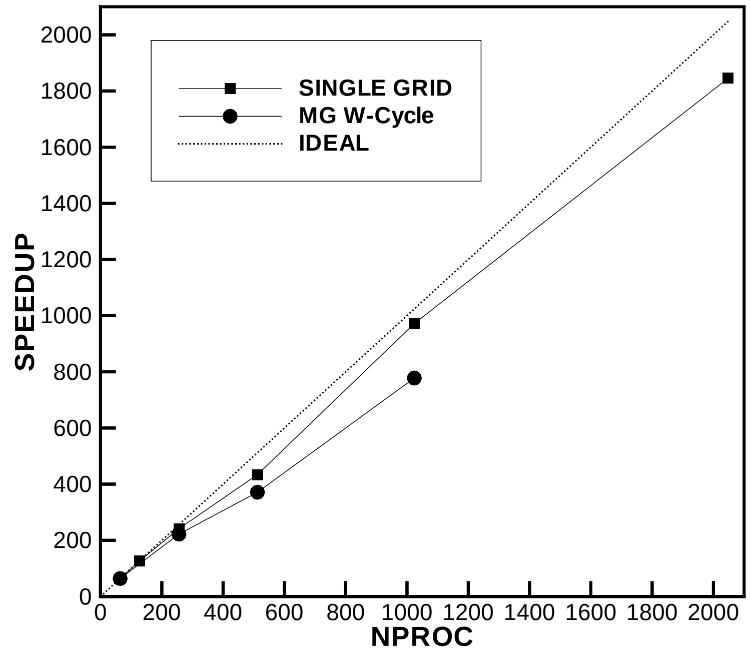


- Good Scalability up to Maximum Number of Processors
  - Larger Problem Size
- Increased Communication for MG Coarse Levels
- MG W-Cycle Always most Efficient Overall

SCALABILITY OF 3M POINT AIRCRAFT CASE



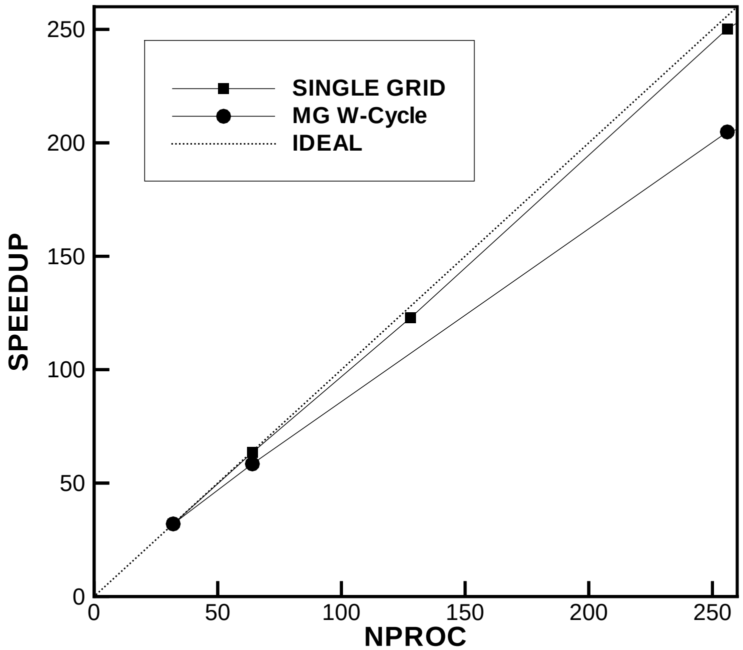
ASCI Blue Pacific (IBM 332 Mhz)



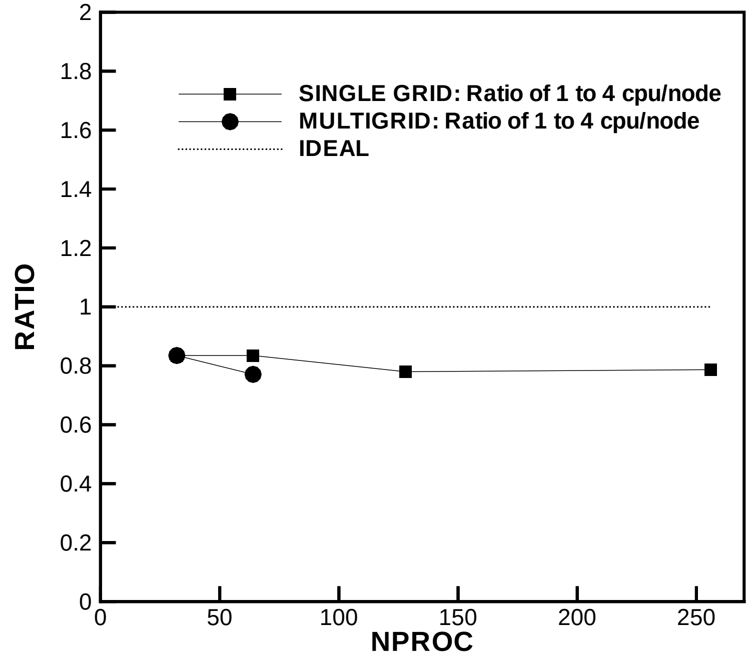
ASCI Red (Pentium Pro 333 Mhz)

- ASCI Blue: Good Scalability up to 256 Processors
- ASCI Red: Good Scalability up to 2048 Processors
- Scalability Improves for Larger Problems
- Increased Communication for MG Coarse Levels
- Coarsest Grid = 1651 Points

SCALABILITY OF 3M POINT AIRCRAFT CASE



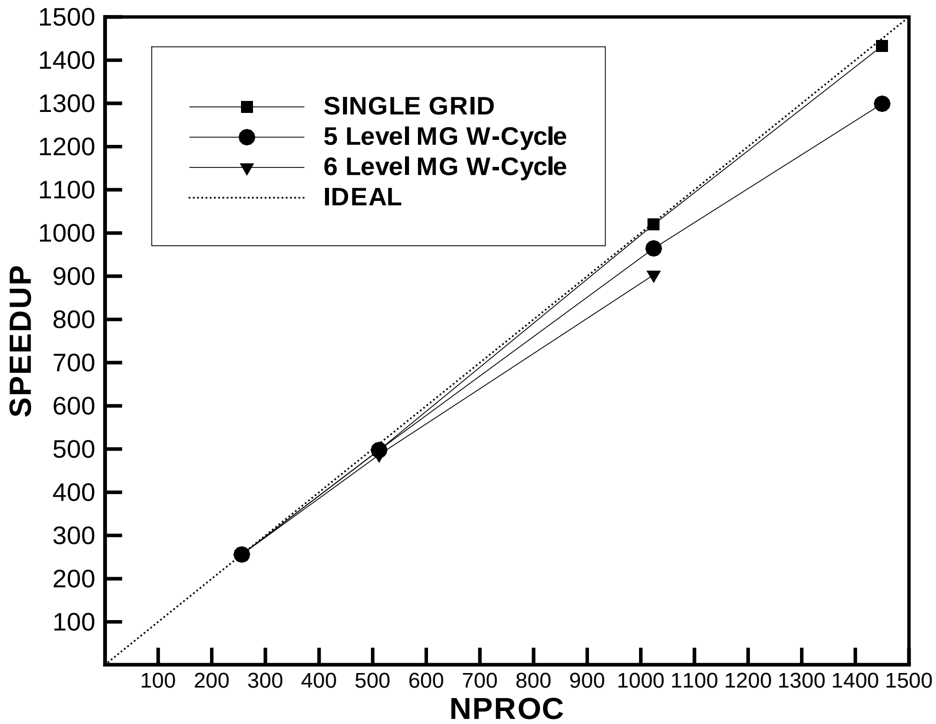
ASCI Blue Pacific (IBM 332 Mhz)



ASCI Blue Pacific (IBM 332 Mhz)

- ASCI Blue: Good Scalability up to 256 Processors
- Slight Degradation due to 4 Shared Memory PEs

SCALABILITY OF 25M PT AIRCRAFT CASE ON T3E-1200E



24.7 Million Pt Case

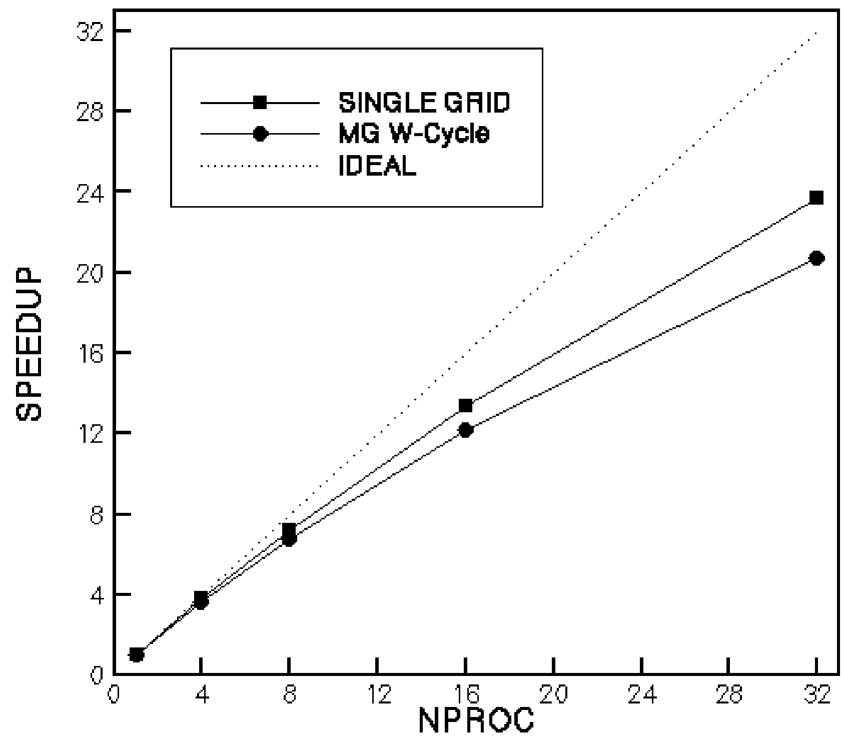
(5 Multigrid Levels)

Platform	No. of Procs	Time/Cyc	Gflop/s
T3E-600	512	28.1	22.0
T3E-1200e	256	38.3	16.1
T3E-1200e	512	19.7	31.4
T3E-1200e	1024	10.1	61.0
T3E-1200e	1450	7.54	82.0

- Dec Alpha 600 Mhz Processors
- Good Multigrid Scalability up to 1450 PEs
- Coarsest Grid = 2208 Points
- 82 Gflops on 1450 PEs (estimated)

ICASE BEOWULF PC CLUSTER

- 32 Pentium II (400Mhz), 8 Gbytes Aggregate RAM
- Fast Ethernet Interconnect
- Total Cost: $50,000

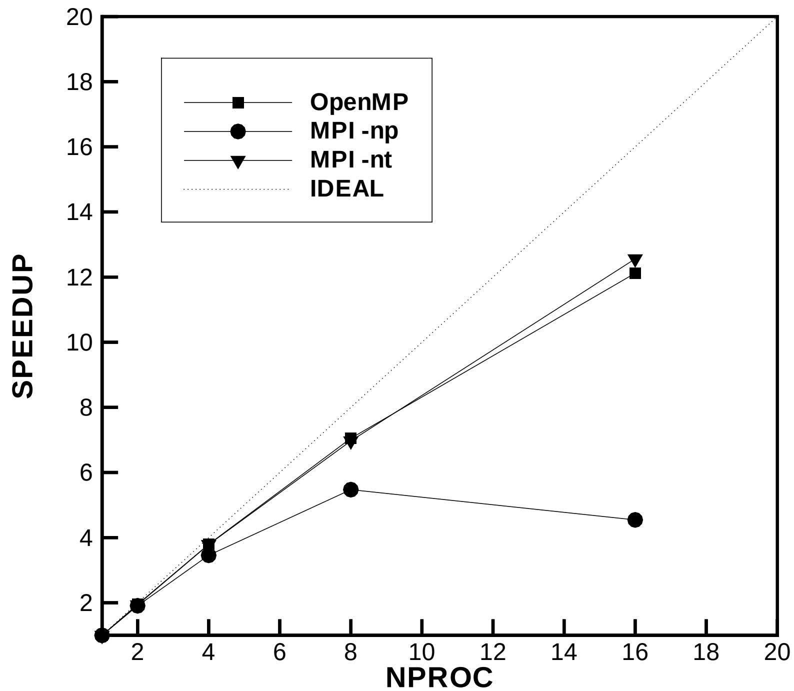


- Scalability of 3D Unstructured Multigrid Algorithm on 177K Grid
- ≈ 1.5 Gflops on Large Unstructured Problems

SAMPLE TURNAROUND TIMES

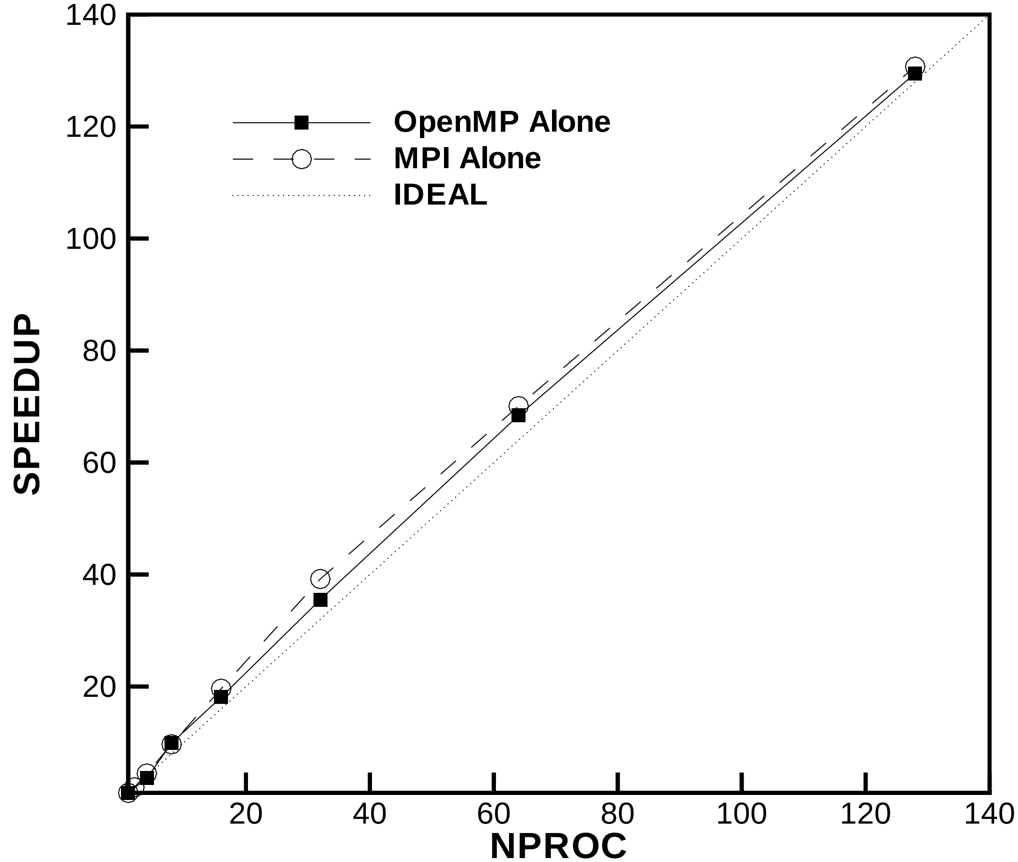
- 3 Million Point Aircraft on ASCI Red (1024 Processors)
  - 21 minutes for 500 Multigrid Cycles
- 25 Million Point Aircraft on T3E-1200E (1450 Processors)
  - 63 minutes for 500 Multigrid Cycles
  - 29 minutes for I/O
  - 9 Gbyte Input File
- Possibility of running over 100 Million Grid Points
- Bottlenecks to be Addressed:
  - Sequential Preprocessing
  - File I/O, Network File Transfer

COMPARISON OF MPI and OPENMP on CRAY SV1



- Vector Machine with Uniform Access Memory
- Two Vendor MPI Implementations
  - MPI -np : Unix Sockets
  - MPI -nt : Shared Memory Communication
- 177K Point Grid, No Multigrid

MPI vs. OPENMP ON SINGLE BOX OF ASCI BLUE MOUNTAIN



- OMP Uses Parallel Initialization (first touch memory placement)
- 3.1 million Point Grid, No Multigrid

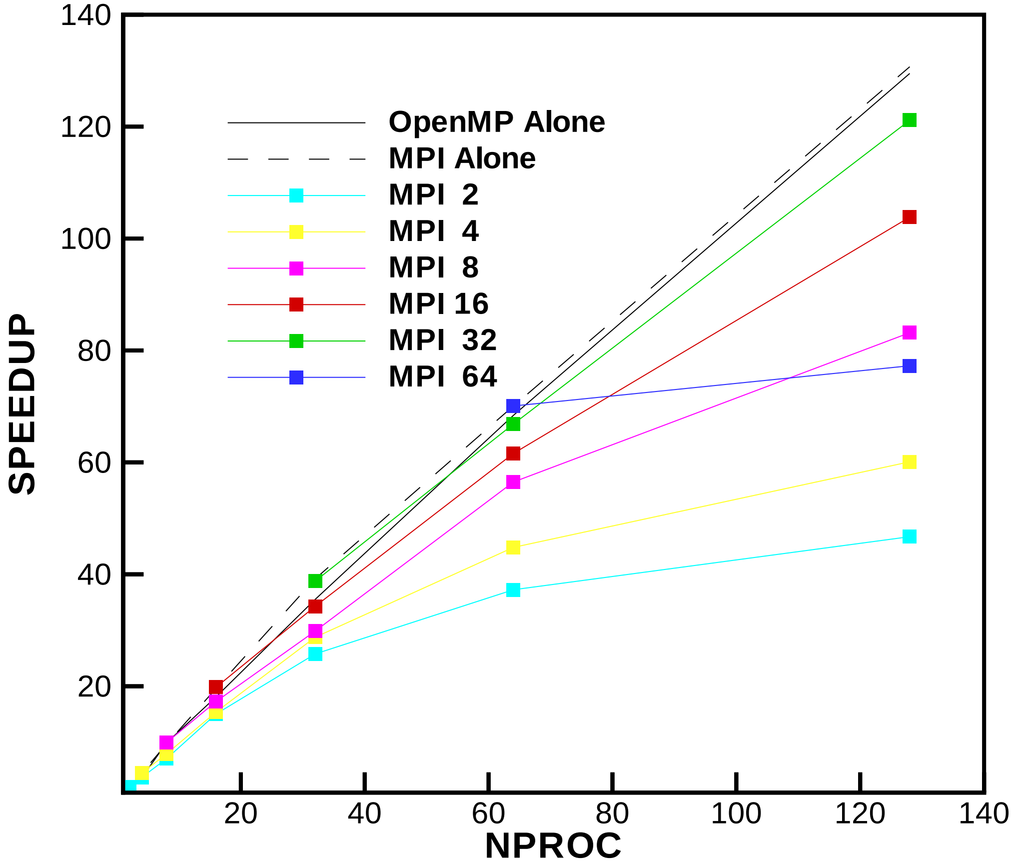
ISSUES AFFECTING PERFORMANCE

- Memory and Processor Placement on SGI Origin
  - Used NASA-SGI Tools for Placement
    - LIBNUMA: mmci, proc, refcnt, mld, mldset, pm, pminfo, numa
  - Requested Processor Placement Not Guaranteed
  - Minimum Memory Placement Page Size
  - Exact Memory Boundaries Cannot Be Prescribed
- Cray SV1 Architecture
  - Physically Shared Memory Architecture
  - Placement not an Issue
  - MPI Performance Dependent on Vendor Implementation

ISSUES AFFECTING PERFORMANCE

- Superlinear:
  - Based on Single CPU Speed
  - 5 Gbytes of Memory Required
  - Off-Processor Memory Access
- Processor Placement Important
  - OS Processor Placement for 8 CPU RUN: 86.9 secs/cycle
  - Explicit Processor Placement for 8 CPU RUN: 60.3 secs/cycle
    - 1 Thread per Memory Node
  - MPI using OS Processor Placement : 61.6 secs/cycle

COMBINED MPI-OpenMP ON ASCI BLUE MOUNTAIN

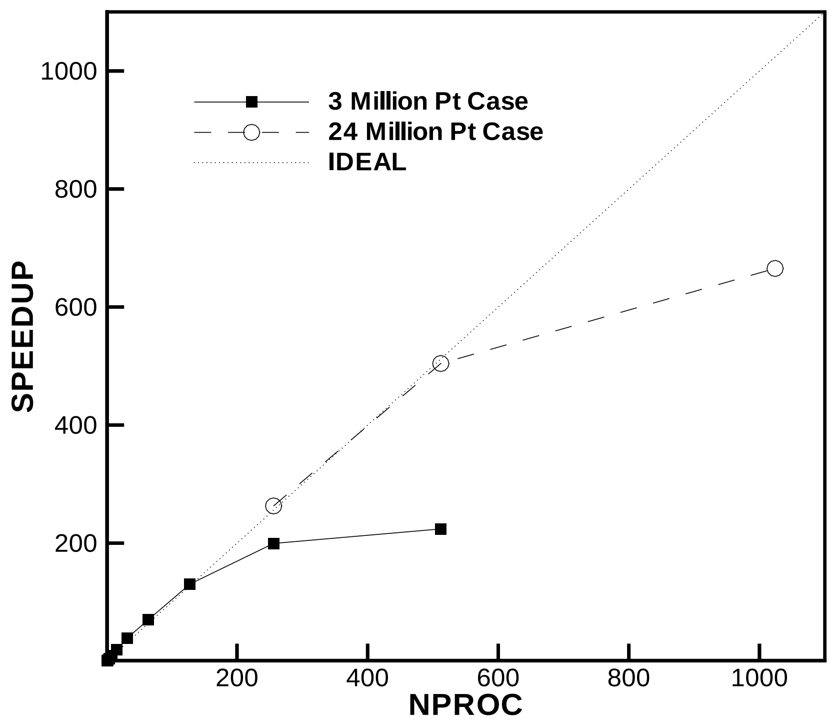


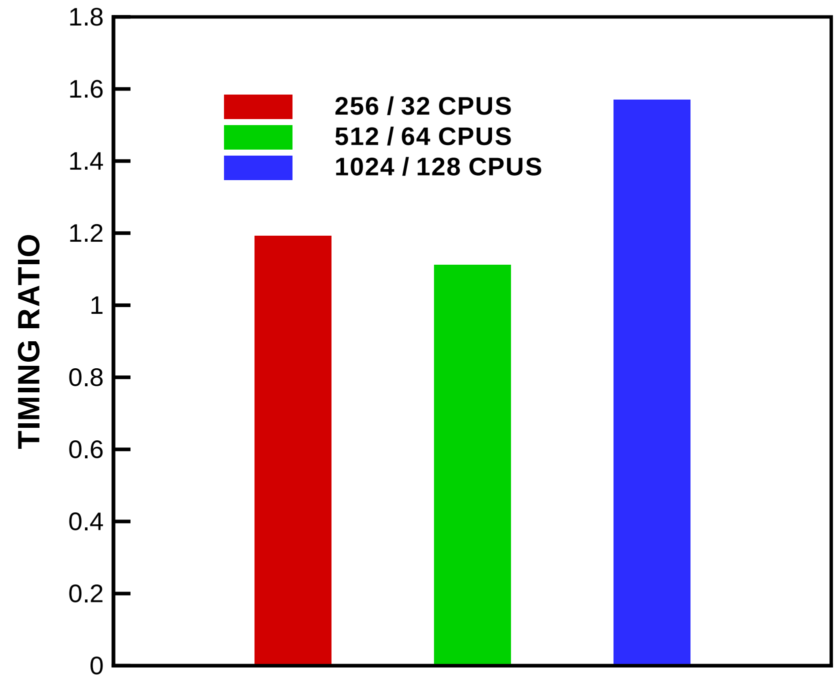
- 3.1 million Point Grid, No Multigrid

MPI/OpenMP PERFORMANCE

- OpenMP and MPI Perform Equivalently on SV1, O2000
  - Validates OMP Implementation
- Combined MPI-OMP Cases Show Degradation
  - Current Origin 2000 MPI Implementation NOT Completely THREAD-SAFE
    - Individual Thread MPI Calls are Sequentialized
    - Degradation Increases with Number of Threads
    - Acceptable for Small Numbers of Threads : Dual CPU Pentiums
- Requested Processor Map Not Always Held
  - Initialized Memory No Longer Local
  - Processes Double up On Single Processor (MPI 64, OMP 2)

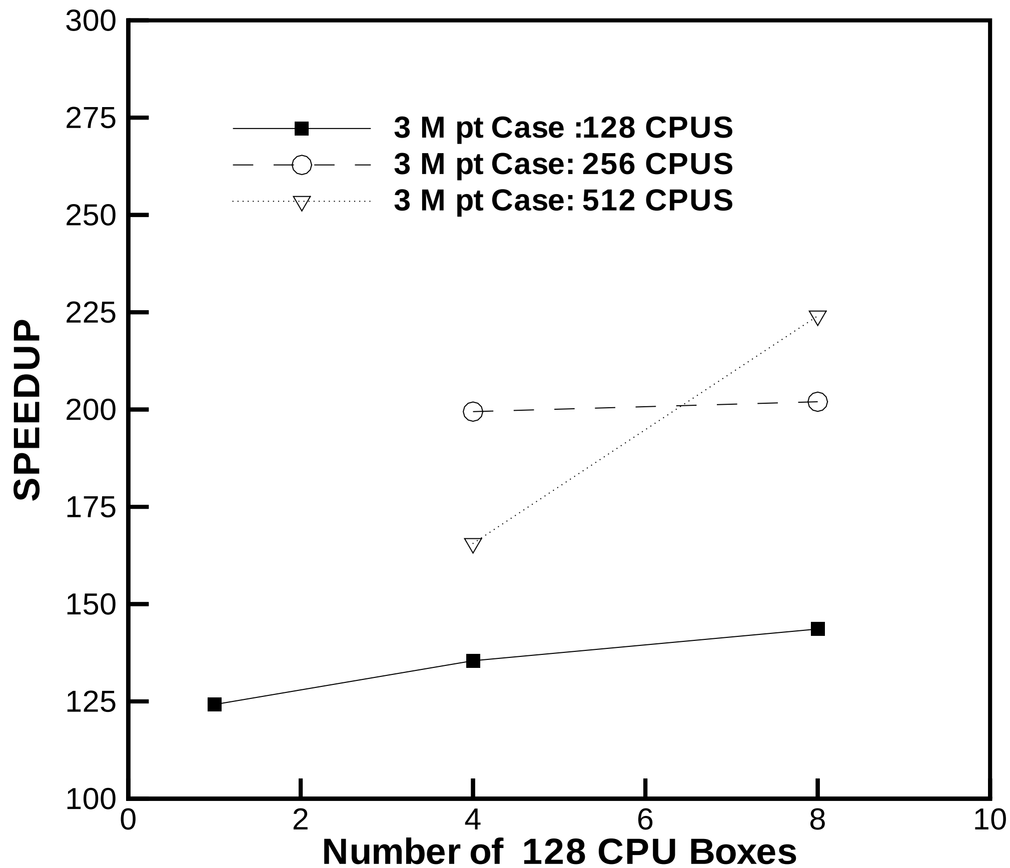
MULTI-BOX PERFORMANCE ON ASCI BLUE MOUNTAIN





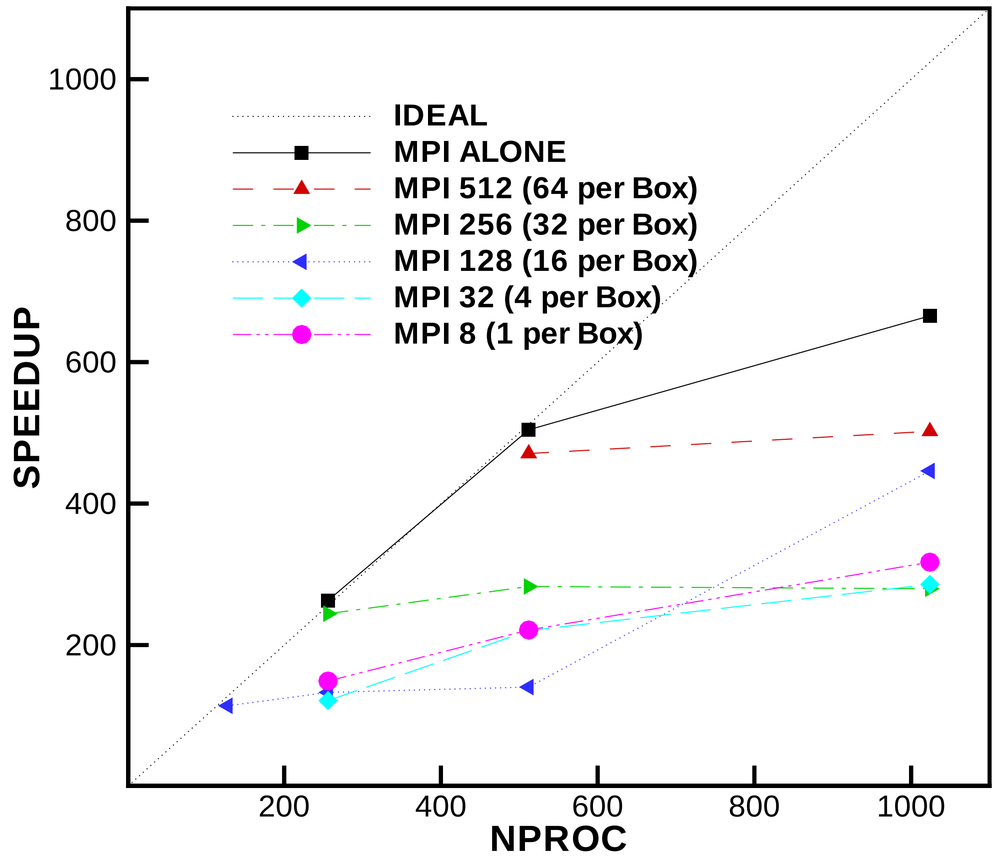
- MPI Alone
- Improved Performance for Larger Problem
- Reasonable Scalability with Problem Size

EFFECT OF THE NUMBER OF BOXES (MPI ALONE)



- Performance Improves Slightly with More Boxes
- Intra-Box Communication is Bottleneck
- Difficulty Maintaining Shared Mem Processor Map

MIXED OMP-MPI ON MULTIPLE BOXES



- Difficulty Maintaining Shared Mem Processor Map
- MPI Not Entirely THREAD-SAFE

CONCLUSIONS

- Current Code Supports Scalar and Vector, MPI and OpenMP
- Best Scalability Obtained on ASCI Red, T3E Machines
- Best Scalability Obtained with MPI Alone

  (even on clustered SMPs)
- OMP and MPI Equivalent on Truly Shared Memory Machines
- OMP and MPI Equivalent on NUMA Machines Provided Memory is Initialized Accordingly and Processes do not Migrate

- Good Combined MPI-OMP Performance Requires:
  - 100 % THREAD-SAFE MPI
  - Ability to Explicitly Map Memory and Processes